REMARKS

Claim 1-5, 11-13, and 15-19 have been amended. Claim 6-10 and 14 have been canceled. No new claims have been added. Claims 1-5, 11-13 and 15-19 are pending. Applicants reserve the right to pursue the original claims and other claims in this application and in other applications.

Claim 14 stands rejected under 35 U.S.C. § 112, first paragraph, as allegedly failing to comply with the written description requirement. Claim 14 has been canceled solely to further the prosecution of the application.

Claims 1-19 stand rejected under 35 U.S.C. § 102(b) as being anticipated by Tock (U.S. Patent No. 5,815,718). The rejection is respectfully traversed.

Claim 1 recites "storing result data of said resolved reference linking to said program in the code entry specified by the instruction." Claim 2 recites "a storing means to store result data of a resolved reference linking to said program in a code data entry specified by an instruction." Applicants respectfully submit that Tock fails to disclose the claimed inventions.

Tock by contrast, referring to Figure 5, states that the:

> invoke method instruction contains a pointer to the method's pointer 511. The offline class loader attempts to resolve this symbolic reference by adding to the method's name a pointer to the method's block. Once the linker has determined the memory layout for the classes, the linker replaces the non-quick format of the invoke method instruction by the quick format which directly references the method. Column 8, lines 8-11 (emphasis added).

In addition, Tock states that the "constant pool contains for each method a method pointer 511 that contains a pointer to the method's name 514 and the method's class name 516. . . . The method pointer 511 is used to symbolically reference a method. This is used in the non-quick format of an invoke method instruction." Column 7, lines 63-64 (emphasis added).

This implies that although the symbolic reference through pointer 511 is resolved by Tock, there remains an additional link that must be made from the invoke instruction to the constant pool entry (pointer 511), to the method's name and finally to the bytecode (i.e., the final destination). This is one difference between the claimed inventions and Tock.

Moreover, another important difference between the claimed inventions and Tock is Tock's partitioning and loading of the program in both RAM and ROM. Tock, for example, states that the "offline class loader tags these methods and data specifying that they are to be stored in a random access memory. All other data is stored in a read-only memory." Column 1, lines 49-52. "By utilizing the offline class loader to partition an application into two address spaces, the amount of RAM utilized by the preloadable module is minimized." Column 1, lines 65-67.

Therefore, Tock specifically teaches an offline class loader that allows a first portion of program methods and data to reside in ROM and a second portion to reside in RAM. However, loading methods and data into RAM increases the Tock system's required start time (this is not alleviated by Tock).

As argued previously, there are additional reasons why the claimed inventions differ from Tock. For example, Claim 1 recites, *inter alia,* the act of "extracting reference data comprising a first and second reference data, said reference

data is used for specifying a location to be accessed in a memory, and resolving a reference using said reference data, said first reference data comprising a resolved class related reference data and said second reference data comprising an index to a resolved field related reference data." Claim 2 recites, *inter alia*, "wherein reference data used to obtain the result data comprises a first and second reference data, at least one of said reference data to specify a location in a memory to be accessed, wherein said first reference data is determined based on class data and said second reference data comprises an index value for one or more field data."

Many modern devices support runtime dynamic loading for object oriented programs in order to permit existing application access to new functionality which may be implemented by the class libraries available during the runtime of program. Traditionally, such functionality is supported by resolving references to class members such as methods and variables during the runtime of the program via a dynamic loader. However, operation of the dynamic loader consumes memory, specifically, random access memory (RAM). In many electronic devices, the available memory includes a read only memory (ROM) and a RAM. In many such instances, the amount of RAM is limited and is therefore a precious resource.

As noted above, Tock discloses an offline class loader that minimizes an object oriented program's usage of RAM by converting an executable module to use both a ROM address space and a RAM address space. To the extent possible, the executable module attempts to rely upon the ROM address space (e.g., for storing static loaded classes and data structures). The offline class loader determines which variables and methods of each class can be stored in the ROM address space, and which methods and variables must be stored in the RAM address space. Column 5, lines 18-21; column 6, lines 1-4. Variables and methods which can be resolved prior to execution are designated for storage in the ROM address space and are re-written with the resolved

address references. Methods which invoke the Java interface or which utilize non-static instance variables are designated for storage in the RAM address space because "the byte codes that implement the interfaces are determined at run time and non-static instance variables are altered for each instantiation of the associated class." Id. at lines 23-26. The offline class loader also performs additional memory optimizations, such as pooling constant data from all classes into a single constant area. The offline class loader outputs one or more class definition files to an assembler, which cooperates with a linker to produce an executable file.

The Office Action asserts that Tock's offline class loader generates an execution module which includes a first reference data based on class and a second reference data which includes an index value for a field (variable). For this support, the Office Action cites Tock column 7, lines 16-17 and lines 45-52. Office Action at pages 8-9. The Office Action further states: "[a]lthough Tock does not use the term "index", (sic) a pointer is used as an index onto the field table. The terms "pointer" and "index" both refer to the use of an offset into a region of data.

The above quoted statement, however, is in fact an admission in the Office Action that Tock does not disclose or suggest the use of an index. As the Examiner is well aware, while indices and pointers are both used to locate specific location of a memory, a pointer specifies an object's location by address, while an index specifies the object's location as count of the number of objections. That is, a pointer would specify information such as "the object is at address 200," while an index would specify information such as "the object is the fifth integer." See also, specification at page 5, lines 14-29.

Accordingly, Tock cannot be fairly stated to disclose, teach, or suggest "said second reference data comprising an index to a resolved field related reference data"

(as recited by claim 1) and "wherein said first reference data is determined based on class data and said second reference data comprises an index value for one or more field data" (as recited by claim 2).

Accordingly, independent claims 1 and 2 are to be allowable over the prior art of record. The depending claims, i.e., claims (3-5, 11-13, and 15-19) are also believed to be allowable for at least the same reasons as the independent claims and on their own merits.

In view of the above amendment, applicant believes the pending application is in condition for allowance.

Dated: January 19, 2005

Respectfully submitted,

By_____

Thomas J. D'Amico
Registration No.: 28,371
Gianni Minutoli
Registration No.: 41,198
DICKSTEIN SHAPIRO MORIN &
OSHINSKY LLP
2101 L Street NW
Washington, DC 20037-1526
(202) 785-9700
Attorneys for Applicant